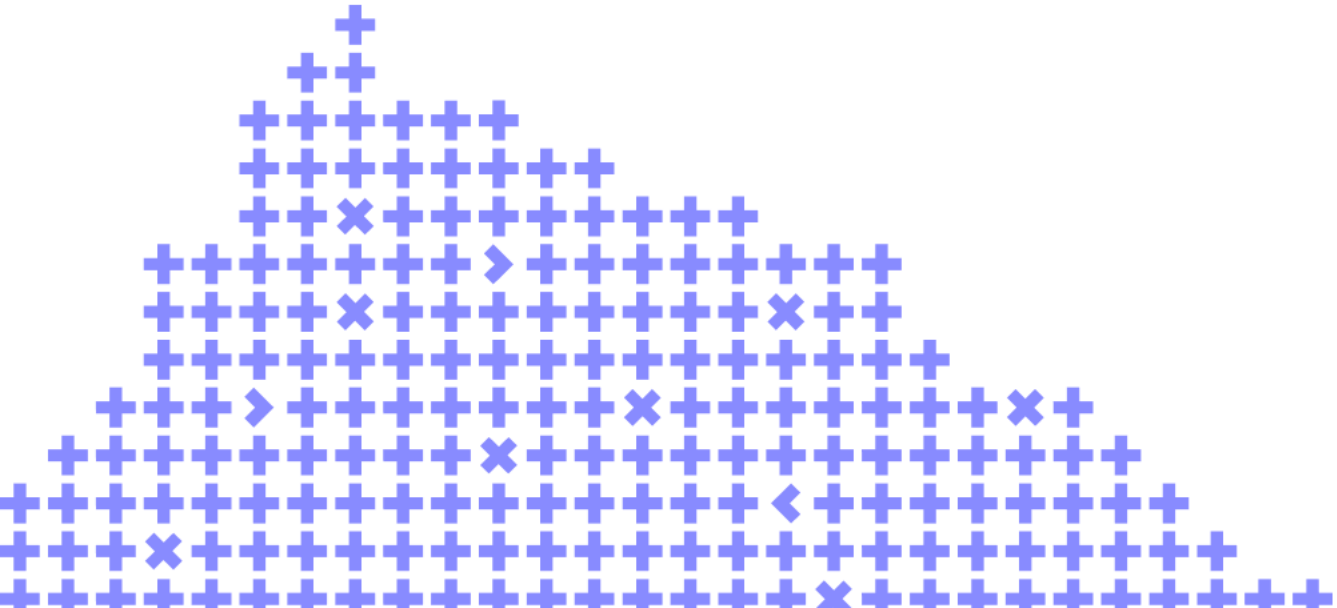
10 mistakes of a (high)load testing in 2022

Evgeny Potapov



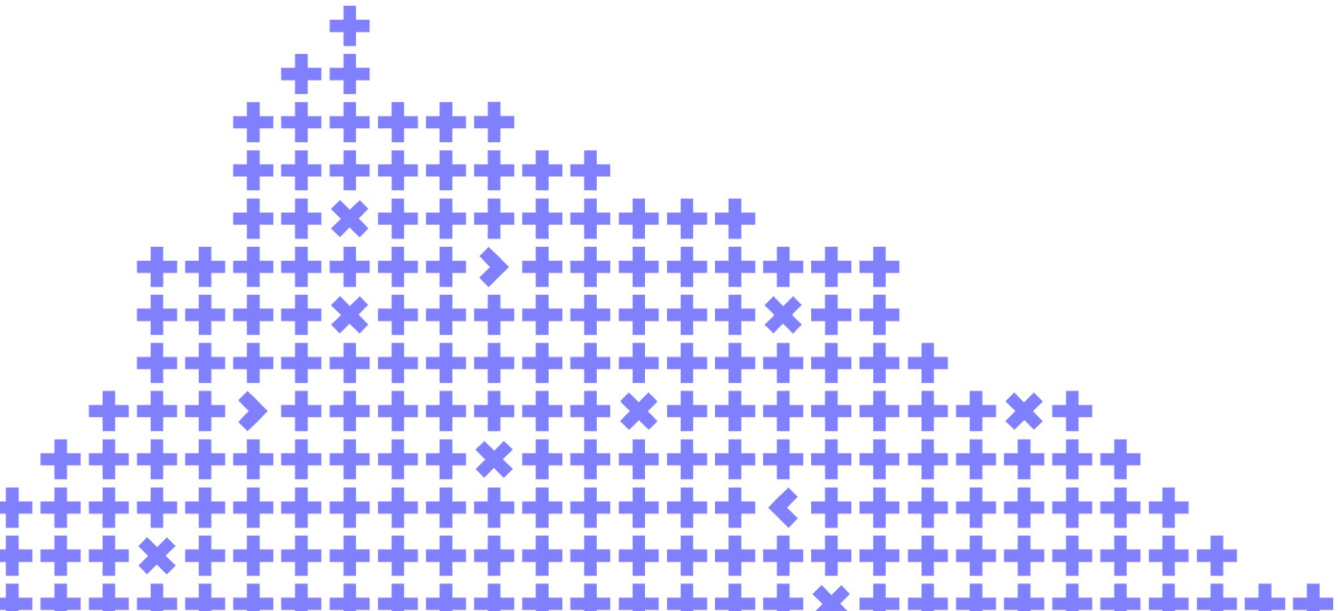
High Load Armenia

Co-organizer

Yandex

10 mistakes of a (high)load testing in 2022

Evgeny Potapov



High Load Armenia

Co-organizer

Yandex

Evgeny Potapov,
CEO DevOpsProdigy, USA

15 years in technical management, SRE, and DevOps on-hand
Regular participant and speaker at Highload++ since 2010
Interests: performance optimization, troubleshooting, fault tol

linkedin.com/in/eapotapov/
eapotapov@devopsprodigy.com
devopsprodigy.com

Defining the problem: Tech businesses in 2022

- Product-led growth
- a large number of competitors
- Retention fight
- Users are accustomed to having new features
- VCs require intensive user base growth

Pressure on development processes

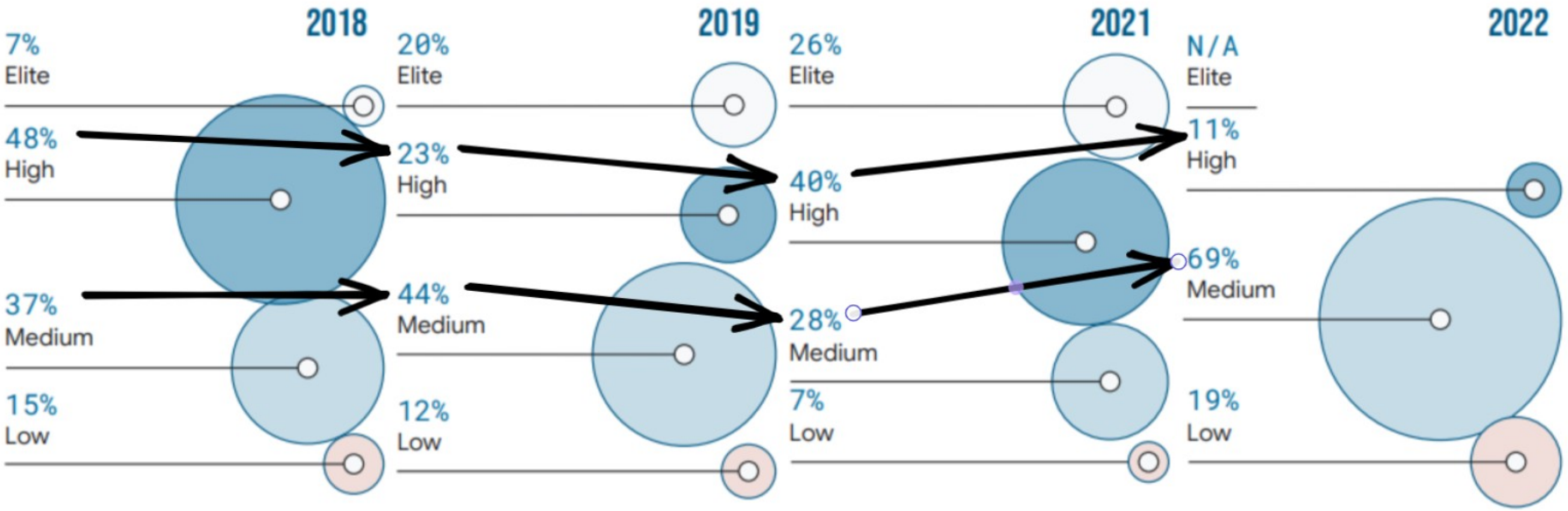
Defining the problem: Tech businesses in 2022

- No more one-time projects anymore (”Office 1997”)
- New features are developed all the time
- Multiple data storages, asynchronous communications, sophisticated queries
- Sophisticated infrastructure built to help with features delivery
- SPAs/XHRs, the frontend is the king
- Sophisticated mobile APIs

State of DevOps 2022

Deployment frequency

Multiple times per day/Once per week



Disclaimer

- Not a real tech person anymore
- I know that load and performance testing and different, but I will substitute
- Focus on the organizational part, not on a tech part
- It might help people, though

1. Not involving performance testing early in the development

“Draw a depressive artwork about not performing performance testing in time”
Author: DALL·E 2

”By the way, we need to do a load testing”

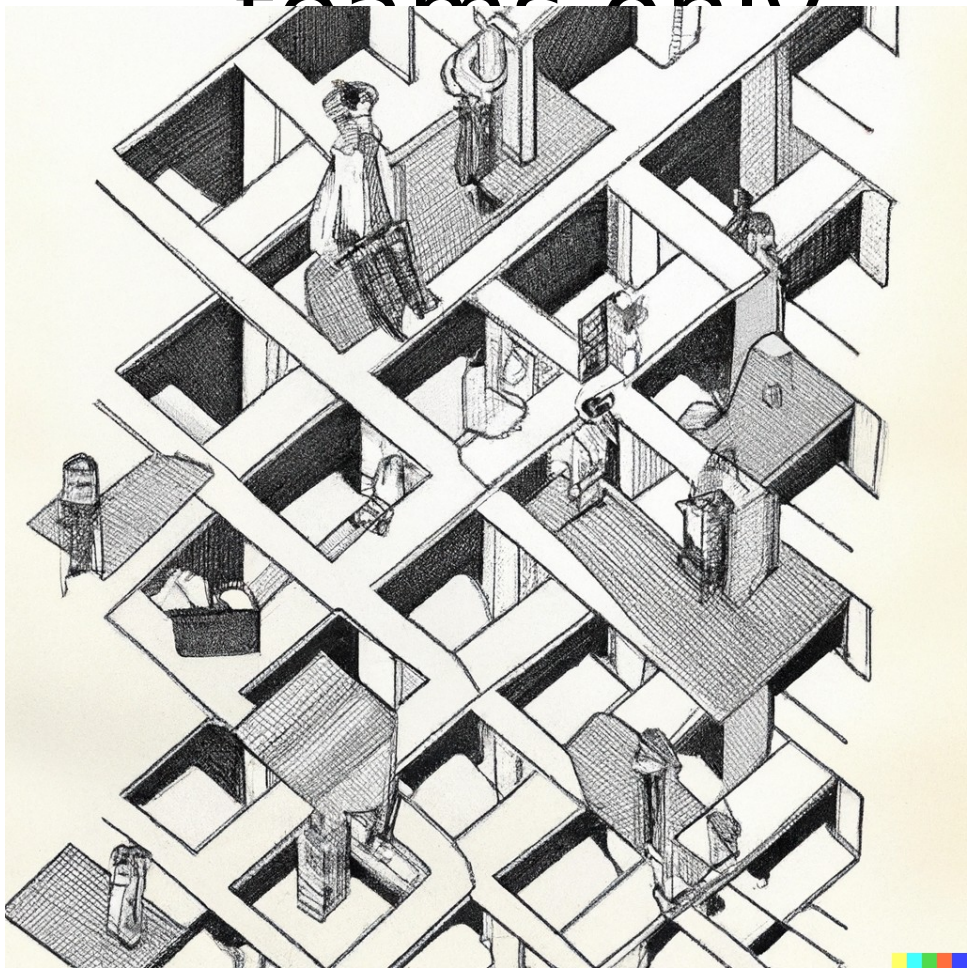
1. Not involving performance testing early in the development process

- No time to implement an adequate performance testing plan
- No time to address performance bottlenecks when it’s easier to fix
- Executing marketing activities that the system wouldn’t be able to handle
- Assigning performance testing tasks to available people, not to experienced people

1. Not involving performance testing early in the development process

- Performance testing is a part of the development process and should be included
- Dedicate experienced people and allocate time for them
- First 24h test will definitely fail, and the second will fail again. Ideally - allocate weeks.

2. Performance testing done by QA teams only



“Draw an isolated team of QA people doing performance testing”
Author: DALL·E 2

2. Performance testing done by QA teams only

“**Essential job duties:**
Develop and maintain manual and automation test cases…..

Create a Performance Test Plan and conduct a Test Plan walkthrough for the project team.

Assist in various types of application testing such as manual, regression, and performance testing using our testing tools through comprehensive test scenarios.”

“**Experience** in performance testing using JMeter.

Understanding of Quality Assurance processes for software application testing

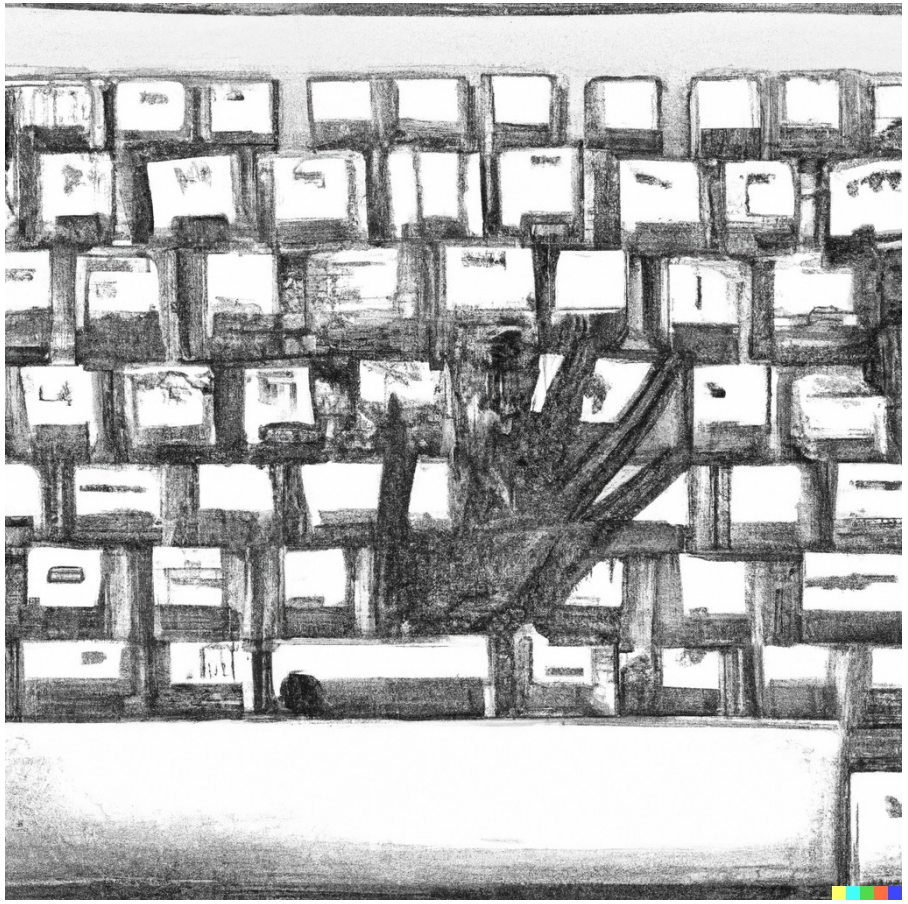
Knowledge of designated testing tool suites such as Quality Center / Azure DevOps / Jira

Knowledge of Microsoft Office suite.”

2. Performance testing done by QA teams only

- **Incomplete or inadequate testing**
  as the QA team may not have a complete understanding of the system or its requirements, which can result in missed opportunities to identify and address performance issues.
- **Lack of collaboration and coordination**
  which can lead to inefficient or ineffective testing, as well as a lack of alignment between the different teams and their goals.
- **Inaccurate or incomplete results**
  as the QA team may not have access to the necessary information or resources to properly evaluate the system’s performance.
- **Delays or bottlenecks in the development process**
  as the lack of communication and coordination between the teams can result in a fragmented or disjointed approach to performance testing, which can lead to delays or other issues.

3. Not having clear performance testing objectives



“Draw a picture of someone defining unclear objectives”
Author: DALL·E 2

”Make sure we can handle 100,000 simultaneous users”

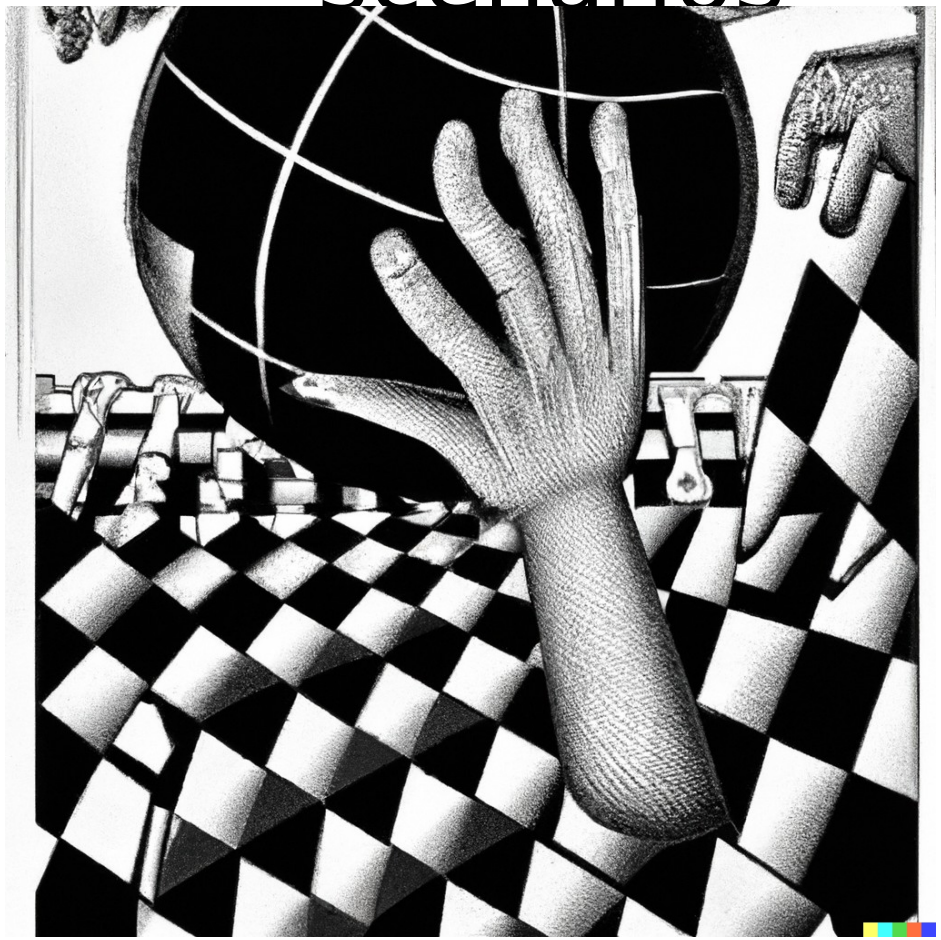
3. Not having clear performance testing objectives

- To determine the maximum number of users that the system can handle without experiencing performance degradation
- To identify and diagnose any potential bottlenecks or other issues that could impact the performance of the system
- To evaluate the system's response time and stability under various workloads and conditions
- To compare the performance of the system to predetermined benchmarks or goals
- To identify opportunities for performance optimization and improvements.

3. Not having clear performance testing objectives

- “We need to know if we can handle 1,000,000 users” – per second? Per minute? Per hour?"
- “What’s going to be the pattern of the traffic?”
- ”How did you get the estimations?”
- “Is this a minimal requirement?”
- “How many users will come from a single campaign? In what period?”

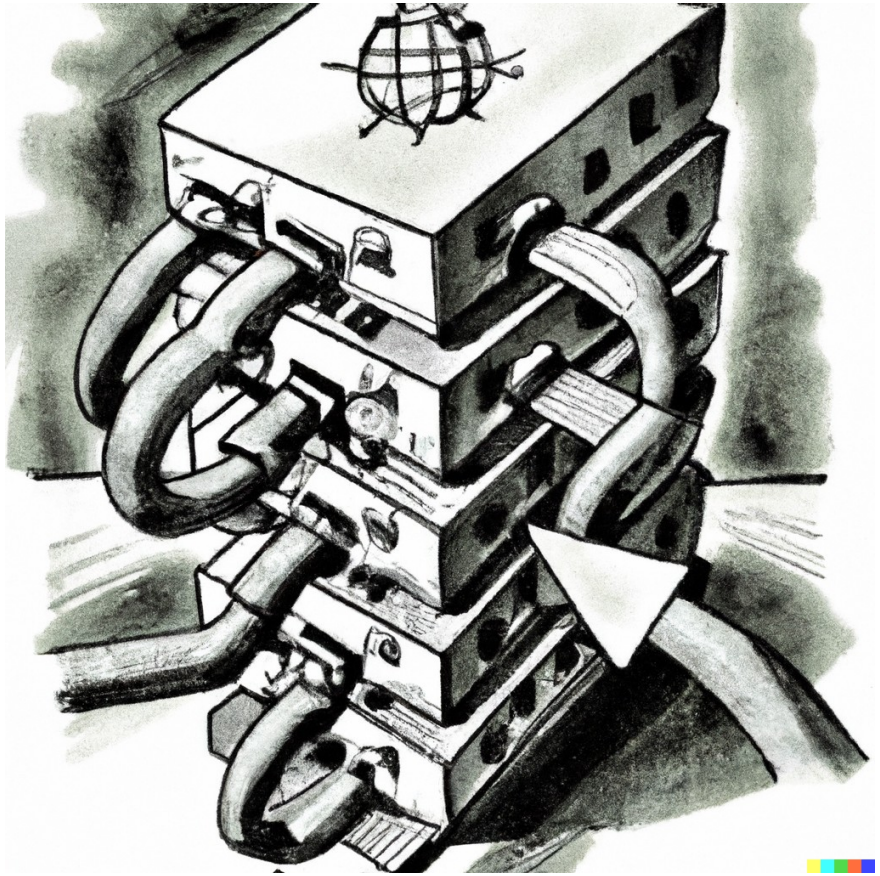
4. Not simulating real-world usage scenarios



4. Not simulating real-world usage scenarios

- Avoiding testing sophisticated functionality (especially billing)
- Testing API endpoints with even distribution
- Avoiding testing sophisticated APIs (GRPC, protobuf etc)
- Avoiding testing microservice interconnections (service bus messaging systems especially)

5. Not adequately preparing the infrastructure

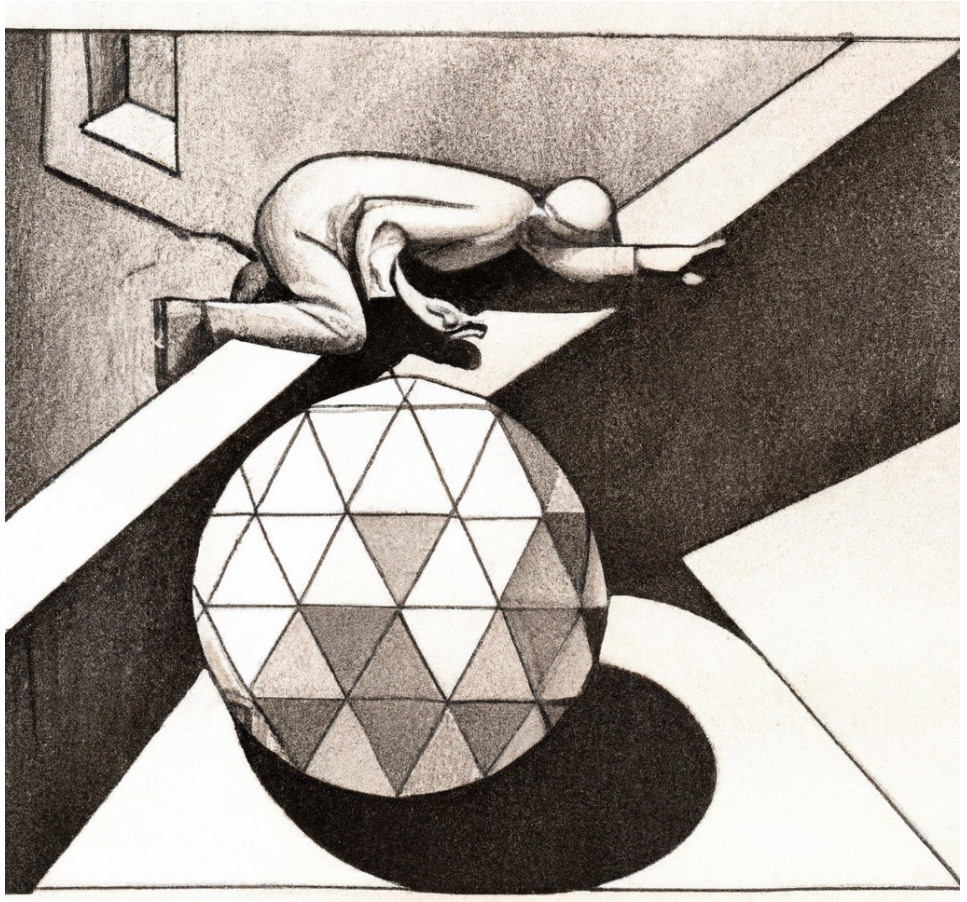


“Draw an adequately prepared infrastructure”
Author: DALL·E 2

5. Not adequately preparing the infrastructure

- Test infrastructure is located within the production infrastructure
- Testing staging environment
- Testing infrastructure is remote
- Testing infrastructure bottlenecks – network throughput especially

6. Common reasons for performance bottlenecks

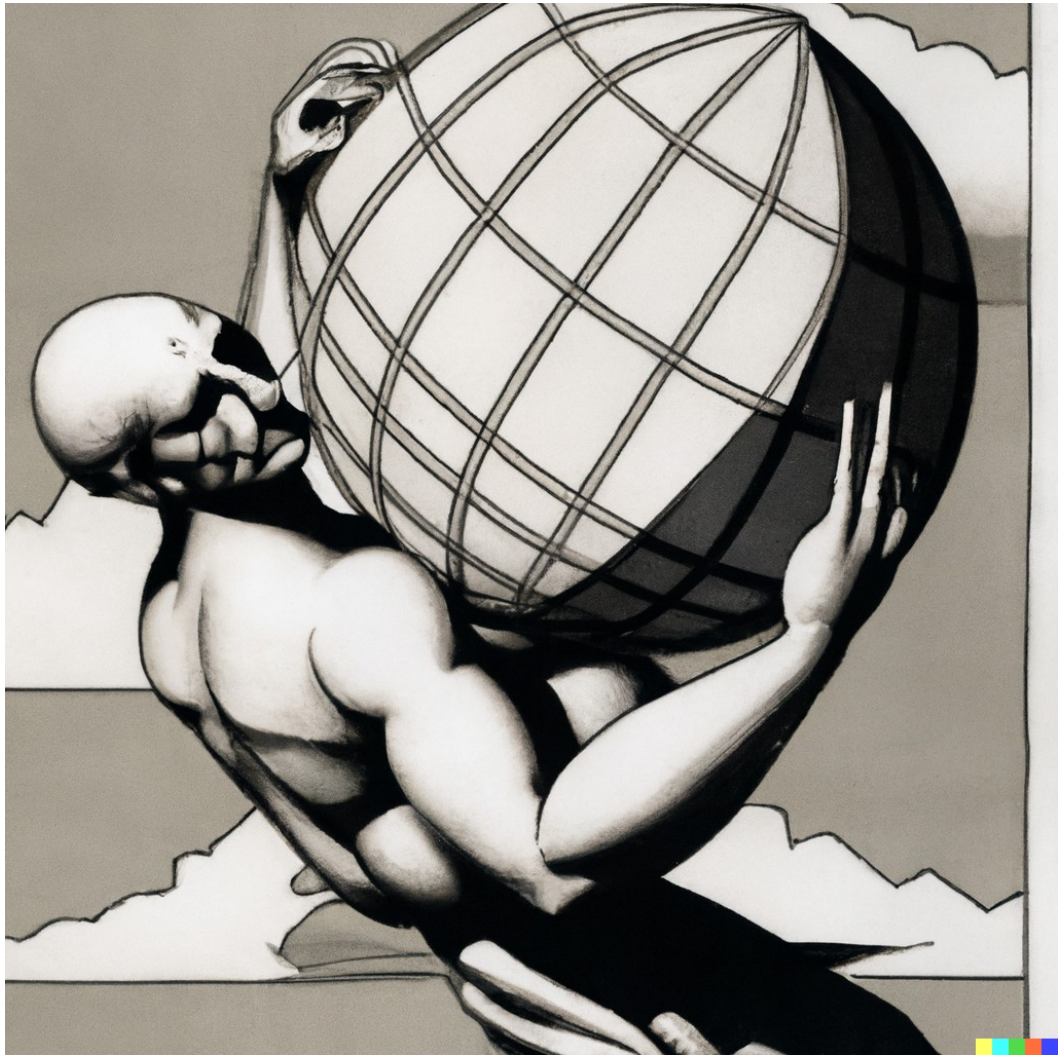


“Draw someone experiencing peak load conditions”
Author: DALL·E 2

6. Common reasons for performance bottlenecks and load test failures

- Data storages bottlenecks
- Network interface throughput problems
- Async queues overflow
- Kubernetes networking issues
- OS-level tuning
- App workers overload

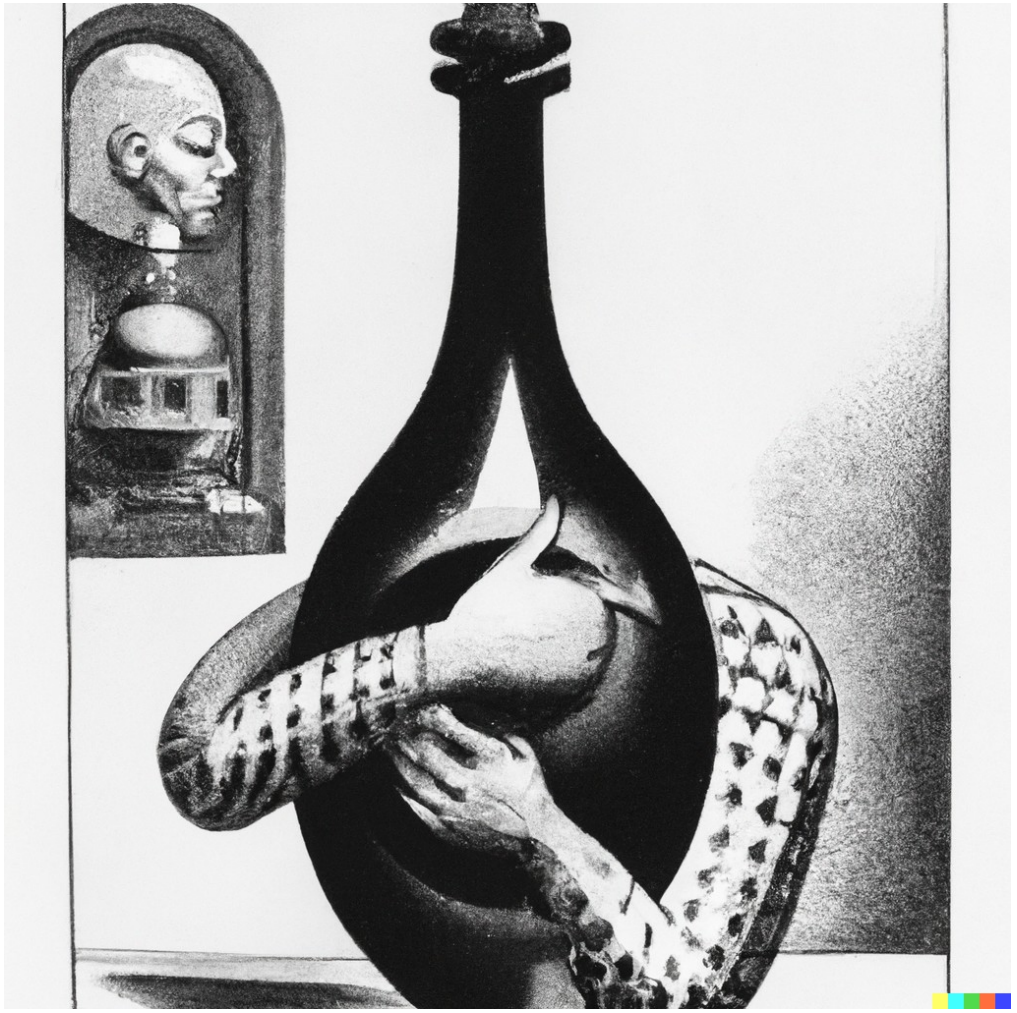
7. Not testing the system for long enough to capture its behavior over time.



7. Not testing the system for long enough to capture its behavior over time.

- Many issues appear only after a long period of the test
  even under the same load
- A failure will occur whenever the execution time for performance testing is changed
- 10 minutes, 60 minutes, 6 hours, 24 hours runs – will take a couple of weeks

8. Performance testing without performance optimization



8. Performance testing without performance optimization

- Of more than 100 performance testing I’ve done, I didn’t see anyone where there were no substantial performance bottlenecks
- Point of failure is not the point of maximum capacity; it’s the point of the next bottleneck
- Sometimes, a simple tuning might add 10x to the application’s capacity
- Proper monitoring is a must

9. Not using appropriate performance testing tools

9. Not using appropriate performance testing tools

- Cloud-based performance testing solutions (eg. k6 cloud)
- grpc, kafka etc: https://habr.com/ru/company/tinkoff/blog/666886/ https://habr.com/ru/company/tinkoff/blog/664674/
- Gatling, JMeter as a standard, but personally I like k6
- Monitoring, monitoring, monitoring!

10. Not accurately analyzing and interpreting performance test results.

10. Not accurately analyzing and interpreting performance test results.

- Of more than 100 performance testing I’ve done, I didn’t see anyone where there were no substantial performance bottlenecks
- Point of failure is not the point of maximum capacity; it’s the point of the next bottleneck
- Sometimes, a simple tuning might add 10x to the application’s capacity
- Proper monitoring in a must

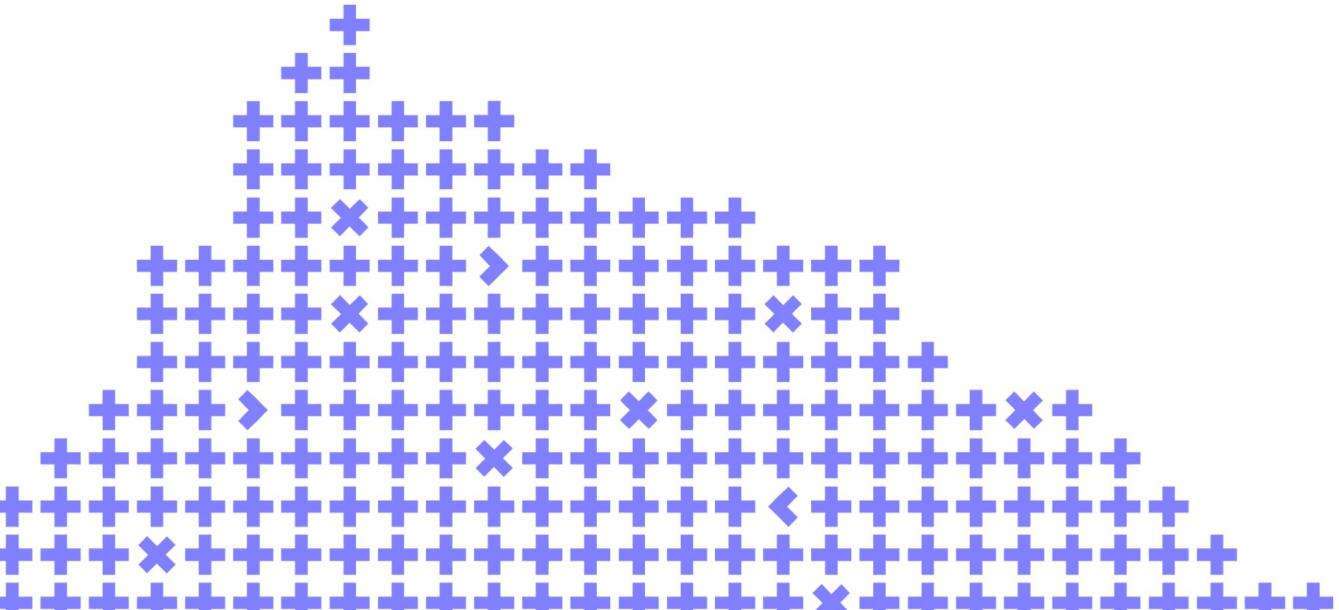
Common problems:

1. Not involving performance testing early in the development process
2. Performance testing is done by QA teams only
3. Not having clear performance testing objectives
4. Not simulating real-world usage scenarios
5. Not adequately preparing the infrastructure
6. Common reasons for performance bottlenecks
7. Not testing the system long enough to capture its behavior over time.
8. Performance testing without performance optimization
9. Not using appropriate performance testing tools
10. Not accurately analysing and interpreting performance test results.

Leave your feedback

linkedin.com/in/eapotapov/
eapotapov@devopsprodigy.com

devopsprodigy.com



High Load++ Armenia

Co-organizer

Yandex